(4)

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER NW-LIS-89-12-01	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) UW VLSI Chip Tester		5. TYPE OF REPORT & PERIOD COVERED Technical 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Neil McKenzie		8. CONTRACT OR GRANT NUMBER(s) N00014-88-K-0453
9. PERFORMING ORGANIZATION NAME AND ADDRESS Northwest Laboratory for Integrated Systems University of Washington Dept. of Comp. Science, FR-35 Seattle, WA 98195		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DARPA-ISTO 1400 Wilson Boulevard Arlington, VA 22209		12. REPORT DATE December 1989 13. NUMBER OF PAGES 20
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Office of Naval Research - ONR Information Systems Program - Code 1513: CAF 800 North Quincy Street Arlington, VA 22217		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this report is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Chip Tester, Apple Macintosh, VLSI, Xilinx, functional testing, CMOS

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

We present a design for a low-cost, functional VLSI chip tester. It is based on the Apple Macintosh II personal computer. It tests chips that have up to 128 pins. All pin drivers of the tester are bidirectional; each pin is programmed independently as an input or an output. The tester can test both static and dynamic chips.

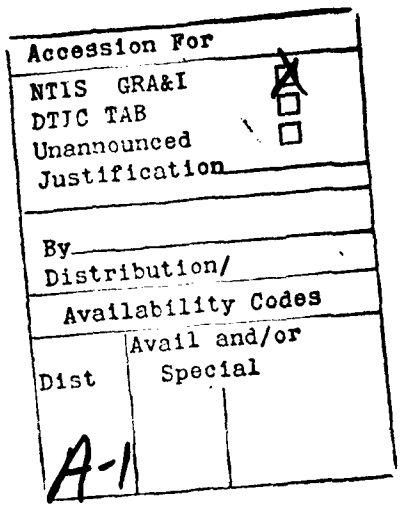
continued on back...

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE
S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

#20 Abstract (continued)

Rudimentary speed testing is provided. Chips are tested by executing C programs written by the user. A software library is provided for program development. Tests run under both the Mac Operating System and A/UX. The design is implemented using Xilinx Logic Cell Arrays. Price/performance tradeoffs are discussed.



UW VLSI Chip Tester

Neil McKenzie

Technical Report 89-12-01

Department of Computer Science and Engineering
University of Washington, FR-35
Seattle, WA, USA 98195
December 1989

UW VLSI Chip Tester*

Neil McKenzie University of Washington

December 1, 1989

Abstract

We present a design for a low-cost, functional VLSI chip tester. It is based on the Apple Macintosh II personal computer. It tests chips that have up to 128 pins. All pin drivers of the tester are bidirectional; each pin is programmed independently as an input or an output. The tester can test both static and dynamic chips. Rudimentary speed testing is provided. Chips are tested by executing C programs written by the user. A software library is provided for program development. Tests run under both the Mac Operating System and A/UX. The design is implemented using Xilinx Logic Cell Arrays. Price/performance tradeoffs are discussed.

1 Introduction

Chip testers are used to test the functional and analog behavior of chips. Chips are tested by delivering a sequence of test vectors to the input pins of the test chip, allowing some time for the test chip to compute its results, then sampling the chip's outputs. Chip testing is an important phase of the VLSI design process. However, because testing is expensive, time consuming, and difficult, it is often not done.

The representation of the test vectors has a significant impact on how easy the tester is to use. For some testers, test vectors are represented in only one way, as an array of binary digits. This representation is difficult to edit and maintain. Furthermore, many chip testers require the user to hand-wire all the pins of the test chip socket. This is inconvenient when the test setup must be changed.

This report describes a design for an inexpensive functional tester, known as the UW Chip Tester. The UW Chip Tester is a combined hardware and software solution to the problem of functional chip testing. This paper first discusses its design goals. Next it gives a physical description of the tester. Next, the hardware design is discussed: cost reduction, design tradeoffs, function, and the actual implementation. Finally, the software is discussed.

*This research was sponsored by Apple Computer, Inc., under grant number ER0030.

2 Design goals

The UW Chip Tester:

- Is cheap and small
- Tests functionality
- Is easy to use
- Requires no hand-wiring in general
- Provides testing and debugging via programming
- Is extensible
- Allows dynamic chips to be tested

The UW Chip Tester has these limitations:

- Speed testing is rudimentary
- No analog testing is done
- Pin drivers are not sophisticated

Our goal is to create a tester that costs an order of magnitude less to build than many commercially available testers. Board area, chip cost and parts count are reduced to the minimum. For the cost of one commercial tester, a laboratory will be able to provide many of our testers. This is a significant advantage in a teaching environment, where there is competition among students for equipment. In the case where a more expensive tester is a necessity, and it is heavily used, our tester can offload the functional testing requirements.

Despite its simplicity, the tester will be easy to use, since all its pins are bidirectional. No hand-wiring will be needed, except for the chip's power and ground pins to provide nigh current. Chip testing is performed by writing and executing test programs that can be tailored to each chip's specific needs. The design is extensible; the number of test chip pins can be easily increased. By adding static RAM to the tester, tests can be executed offline from the host CPU.

The tester has its limitations. Speed testing is limited to measuring the propagation delay between the inputs and the outputs of the test chip. The tester uses CMOS pin drivers and latches; both CMOS and TTL chips can be tested. The CMOS pin drivers drive a limited amount of current. The user has no control over switching characteristics such as slew rate. Each test pin drives and tests a binary value. Analog testing is not performed by the tester, but it is simple enough to use an oscilloscope to perform some analog analysis.

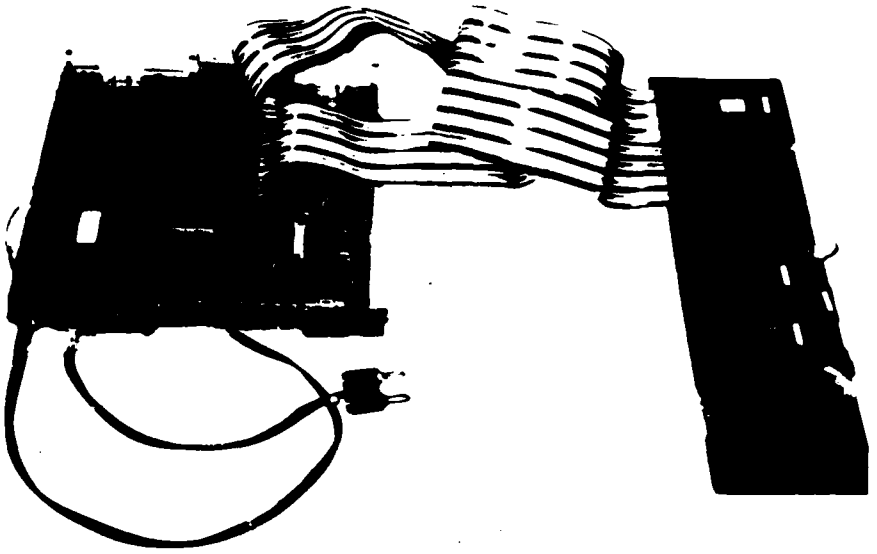


Figure 1: The UW Chip Tester prototype

3 Overview

3.1 Physical description

The UW Chip Tester system consists of a Macintosh II, a NuBus interface card, the tester unit, and cables. Figure 1 is a photograph of the prototype of the tester.

The tester system uses a NuBus interface card to provide a parallel port. Currently, the Adex MacProto card is being used. The MacProto card provides a 32-bit data bus, an address bus, and a 7-bit control register. There are also handshaking signals for CPU read and write. These signals are brought out by cables to the tester unit.

The tester unit consists of a PC board, connectors, sockets for the chips under test, and a power supply. The tester unit has wirewrap pins for jumpering power or ground directly to the test chip, or for oscilloscope probes. Seven Xilinx XC3020 chips are used: one for the controller, and six for the data path. The UW Chip Tester can test chips with up to 128 pins. Different chip sockets are handled by having different personality boards that plug

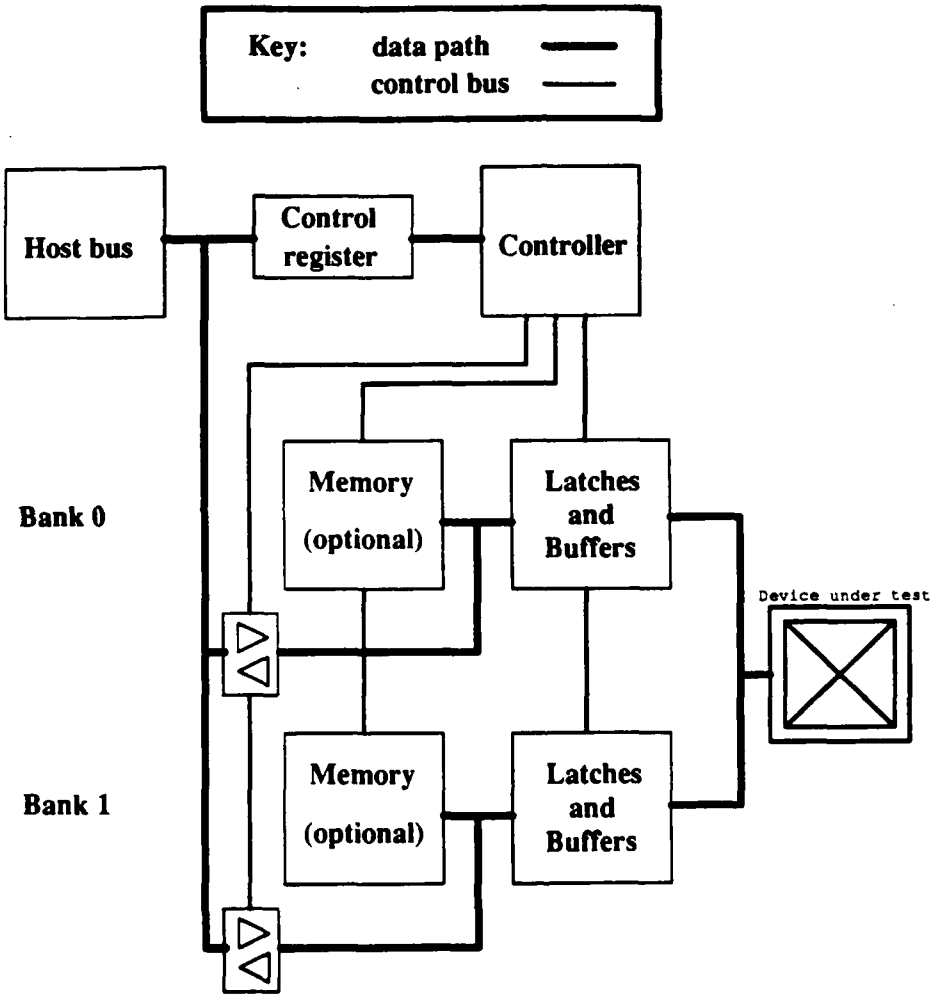


Figure 2: Block diagram for UW Chip Tester

into the tester unit.

Figure 2 is a block diagram of the tester unit. The tester hardware consists of a controller module and data path modules. A single data path module is called a *bank*. Each bank control 64 test chip pins. More banks can be added to the design in order to increase the number of test chip pins.

We hope to make the UW Chip Tester available to universities, as a kit with an unstuffed PC board, software, and documentation. The tester unit stuffed costs about $500. The entire system can be assembled for less than $1000.

3.2 Software interface

A chip test is represented as a custom software program. The test is performed by executing the compiled form of that program. Programming provides high-level flow control and structure. Pins of the test chip are represented as data structures in the test program. This representation is more expressive and easier to create and maintain than an array of binary digits.

The UW Chip Tester is based on the Apple Macintosh II. Test programs may run under either MacOS or A/UX. The test package is currently implemented as a C library. Test programs are written in C and then linked to the library.

4 Data path tradeoffs

This section describes the tradeoffs between different strategies for implementing the tester's data path. Price and performance of each strategy are discussed. The UW Chip Tester's implementation is was chosen as the best compromise of low cost and good performance.

4.1 Simple scheme using single buffering

One important design goal is software control of the pin drivers, such that any pin can be configured as an input or output on-the-fly. This goal can be realized by using three bits of state per pin of the test chip. One bit stores the input value, one bit stores the sampled output value, and one bit controls a three-state buffer. See Figure 3.

This scheme can be driven by a parallel port. Extensibility is provided by adding parallel ports as needed. The sample clock is also generated by a parallel-port signal. The entire output vector is latched simultaneously.

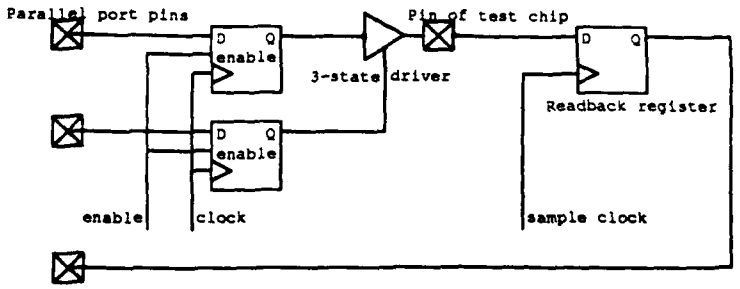


Figure 3: A simple scheme using single buffering

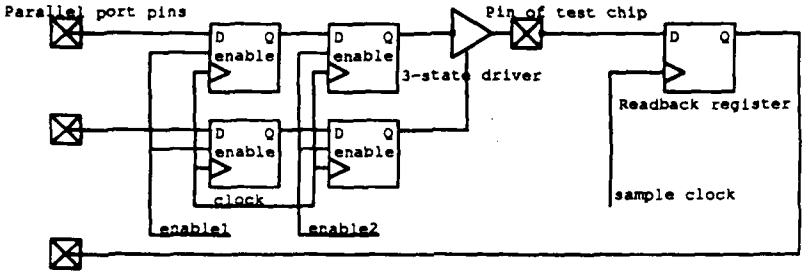


Figure 4: Using double buffering

The advantage to this scheme is that it is simple and cheap. The drawback to this scheme is that a large test chip will need more than one parallel port. This means that the test chip's input vector is not guaranteed to arrive simultaneously. In other words, there can be a large skew between signals, on the order of microseconds. We considered this to be a severe restriction, and did not implement the data path in this way.

4.2 Double buffering

In order to limit the skew, an extra layer of buffering is used. This design requires five bits of state per pin of the test chip. The first column of input state bits are loaded asynchronously, and then the entire column is propogated simultaneously to the inputs of the test chip. See Figure 4.

The advantage to this scheme is that it is still reasonably cheap. One drawback to this scheme is that speed testing is limited by the number of parallel ports needed. Each parallel port may need to be accessed.

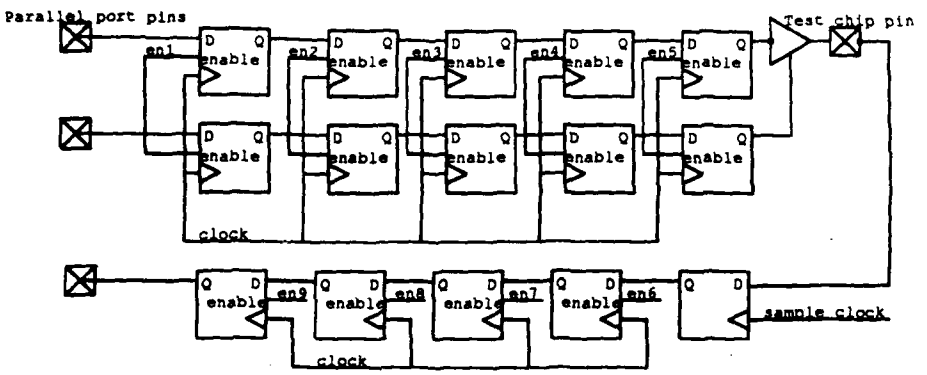


Figure 5: A four vector queue

However, if only one bit changes (e.g. a clock signal), only one parallel port will need to be accessed. The larger the number of parallel ports that need access, the slower the vector cycle time. This can affect the ability of the tester to do true speed testing.

4.3 Queue

In order to provide true speed testing, input vectors must arrive at the test chip at the same interval as the sample clock. One method to implement this is a first-in-first-out queue, composed of a series of registers. Queues are needed to store both the input and output vectors. Figure 5 is an example of a queue four vectors deep. A four vector burst is sufficient for testing CMOS chips that use two-phase clocking, and are static when no clocks are active. The expense of this extra functionality is linearly proportional to the size of the queue. Another method that we studied used an off-the-shelf FIFO chip. This solution was relatively expensive and we did not pursue it.

4.4 Local memory

Another way to perform speed testing is by using local memory to store the test vectors. This is a standard tester design. Vectors can be moved between the host's memory and the tester's local memory when the tester is not actively testing chips. The test is set up by the host, then it is executed offline from the host. The test is halted, and then the results are read back by the host. The test interface is batch-oriented rather than interactive when using local memory. See Figure 6. The local memory also gives the tester the ability to test dynamic chips (see the following subsection).

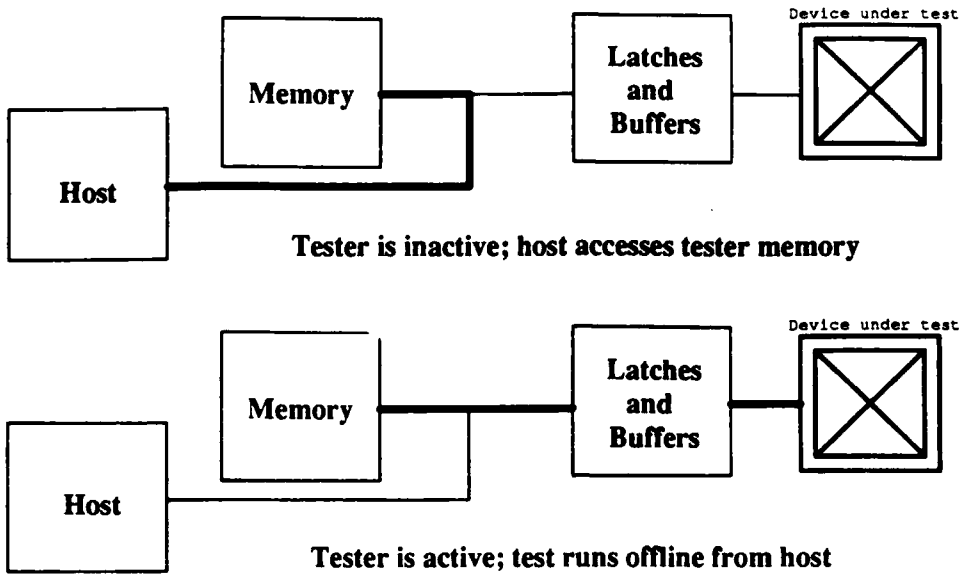


Figure 6: Local memory access

The speed of the tester is a function of the access speed of the memory chips. Static RAMs are preferred over dynamic RAMs because they have faster access times. Local memory subsumes the need for the queue. The queue's advantage is that it is still the fastest method, but its expense is linear in the size of the queue. The local memory solution is slower, but its expense is less than linear in the size of the memory. If a large number of vectors is needed, the local memory solution is the best one.

4.5 Dynamic chip testing

Dynamic chips are chips that have dynamic state. A familiar example is a dynamic shift register. Its internal charges are volatile. If the shift register is not clocked at a certain rate, it will lose its information.

Dynamic chip testing is similar to speed testing. Speed testing requires a *minimum* timing requirement; the speed is increased until the chip fails. By contrast, dynamic chips have a *maximum* timing requirement; if its signals change too slowly, the dynamic chip will fail. Dynamic chips are common, and it is important to be able to test them.

Dynamic chip testing is accomplished either by using a sequencer and local memory to run a test offline from the CPU, or using the CPU such that it is dedicated to the testing

task. Dynamic testing in the latter case may fail under a multitasking operating system like Unix. Other processes may take away the CPU for too long a time. Moreover, it may be impossible to get consistent results, since context switches happen asynchronously.

5 UW Chip Tester implementation

5.1 Cost and space analysis

Our initial strategy for implementing the tester was with small-scale integrated circuits. Since we wanted to have CMOS-compatible pin drivers, we would use the HC family of SSI chips. We analyzed the cost-per-pin-driver of the test chip using the double-buffering scheme. Each pin requires a minimum of five flip-flops and a three-state driver. This is equal to 5/8 of a 74HC374 and 1/4 of a 74HC125, or roughly 7/8 of an HC part. These parts cost about $2 each; the cost per pin is roughly $1.75. For 128 pins, the parts cost is about $225.

The parts count for 128 pins is large. It takes 80 74HC374's and 32 74HC125's; that's 112 chips for pin drivers alone. We decided that this was unacceptable. The PC board needed would be huge.

In order to reduce the chip count, we looked at higher-density chips, such as Programmed Array Logic chips. The highest density PAL we found was the 22V10. We saw that each 22V10 could control two pins of the test chip. For 128 test chip pins, 64 of these PALs would be needed. Each 22V10 costs between $10 and $15. Although the parts count is halved, the parts cost would at least triple with respect to the SSI solution. We judged this to be unacceptable.

To achieve higher density, we considered custom logic chips such as gate arrays. Much higher density would be realized. However, custom chips require greater economies of scale to make them cost-effective. Custom logic chips have a large non-recoverable engineering cost. Also, they have limited availability since they cannot be bought off-the-shelf. Therefore, we decided not to use custom logic in the tester.

Fortunately, we found a technology that implements the tester well. We chose the Xilinx Logic Cell Array to implement much of the UW Chip Tester design.

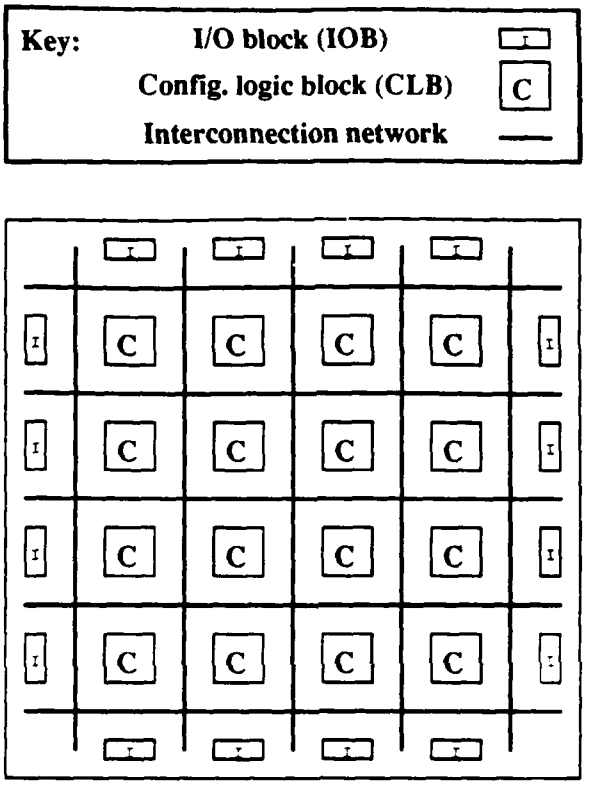


Figure 7: A simplified floor plan of the Xilinx LCA

5.2 Xilinx Logic Cell Arrays

LCA technology has a number of features that are favorable:

- Higher gate density than PAL technology
- Flexibility
- Reprogrammability
- Low cost per gate
- CMOS implementation; low static current

An LCA consists of an array of Configurable Logic Blocks (CLBs) in its interior, a ring of Input Output Blocks (IOBs) around the perimeter, and a network of interconnection

paths. See Figure 7. In the XC3000 family of LCAs, each CLB contains two flip-flops and it computes any function of 5 Boolean variables, or two functions of 4 variables. The IOBs can be programmed as inputs, outputs, or bidirectional. Refer to [1], chapter 2, for a more complete treatment of the LCA's internals.

The LCA used in the UW Chip Tester is the XC3020, which has 64 IOBs[1] and 64 CLBs. The 3020 costs about $45. The UW Chip Tester design uses this LCA to drive the pins of the test chip. Each 3020 controls and samples 22 pins of the test chip. The cost-per-pin of the tester is about $2. This compares closely with the SSI parts cost, but has a much smaller area requirement.

LCAs are programmed in-circuit at power-on time. The bit stream that represents its circuit is clocked into the LCA in bit-serial fashion, using a two wire interface. This is LCA Slave Mode, as it is described in [1], page 2-20.

The LCA bit streams are stored on disk on the host computer. An application program is used to initialize the LCAs. The initialization takes less than 100 milliseconds; this is negligible compared to the time it takes to launch an application. Different versions of the LCA circuits can be made available on disk to make the overall circuit perform differently. This is helpful for providing field upgrades.

When an LCA is uninitialized, its outputs are configured such that there is a passive pullup to the power rail Vdd. This allows the LCA's I/O pins to be directly connected to Vdd or ground without hazard to the LCA at pre-initialization time.

5.3 Multiplexing

One design decision that was motivated by the LCA architecture is the use of multiplexing. The LCAs are pin-limited, so it is advantageous to reduce the number of LCA I/O pins needed that do not drive the test chip.

The solution is to reduce the number of parallel-port pins. Figure 8 demonstrates this; the parallel-port pin is multiplexed three ways among input, output and buffer control. Each flip-flop requires a separate enable signal, which is computed from address bits or control register bits.

Furthermore, the data paths for more than one pin of the test chip can be multiplexed together. Figure 9 demonstrates how one parallel-port pin is multiplexed six ways in order to control two test-chip pins.

Another benefit of multiplexing is that tester systems with local memory (see Section 4.4) require fewer RAM chips. The information in local memory is interleaved. In the case

[1]Six IOBs are not bonded to physical pins in the PLCC-68 version of the 3020; therefore there are effectively only 58 IOBs.

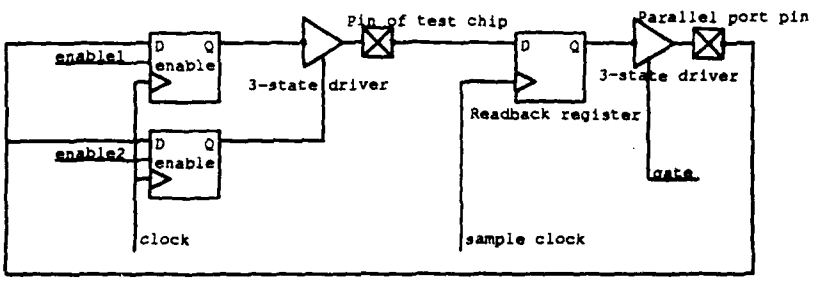


Figure 8: Multiplexing the parallel-port pins

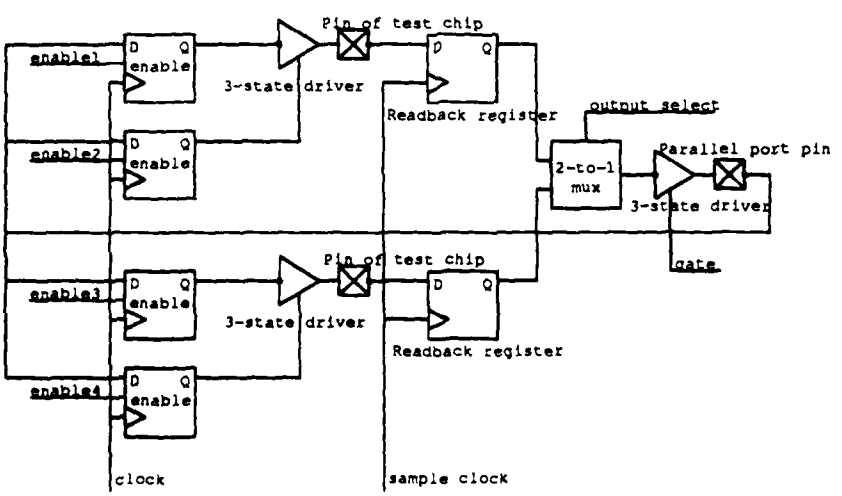


Figure 9: Multiplexing two test-chip pins

of the six-way multiplexing strategy, six words of memory are used for every test vector, according to the following table:

```
Word 0:     pin 1 data
Word 1:     pin 1 enable
Word 2:     pin 2 data
Word 3:     pin 2 enable
Word 4:     pin 1 result
Word 5:     pin 2 result
```

One drawback of multiplexing is that less parallelism is available; more memory cycles are needed to execute a single test vector. This defeats the ability to perform speed testing at the full speed of the memory chips.

5.4 Extensibility

To increase the number of test chip pins, the entire data path circuit can be replicated. One instance of this circuit is known as a *bank*. In the UW Chip Tester, each bank requires three XC3020 chips. Together, they control 64 test chip pins. The current implementation has two banks which yields 128 test chip pins.

Each bank is mapped to the same physical address space. Banks are selected by bank-select bits in the control register. Only one bank can be accessed at a time by the host computer. However, when the tester is actively testing a chip, all banks run in parallel.

5.5 Data path implementation

The UW Chip Tester uses the least expensive scheme for the data path that has the capability to test dynamic chips. This is done via the double-buffering scheme with six-way multiplexing. Each parallel-port pin drives two test-chip pins. Figure 10 describes the circuit for controlling two pins of the test chip. This circuit is a macro which is instantiated eleven times inside each data path LCA. This circuit is CLB-limited. CLB utilization exceeds 90 per cent in the XC3020.

Local memory is optional in the UW Chip Tester. Each bank may contain four static RAM chips. The implementation uses static RAM chips that have 8 data pins. Any size of static RAM from 2K×8 to 32K×8 may be used. All RAM chips in the tester must be the same size, since all of them are accessed in parallel during offline testing. The number of vectors is the size of a single RAM chip divided by six. E.g. if 32K×8 RAMs are used, then the number of vectors is 5461. The PC board layout is the same whether or not the memory is installed.

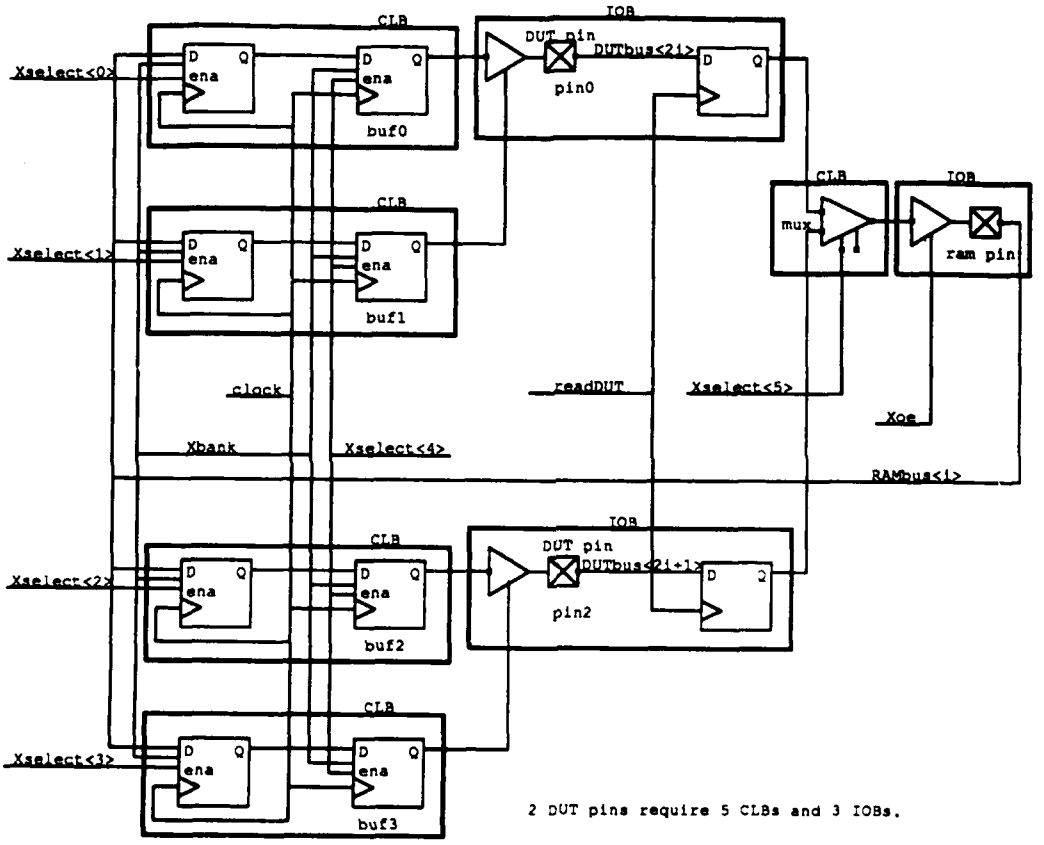


Figure 10: Data path macro showing CLB and IOB usage of LCA

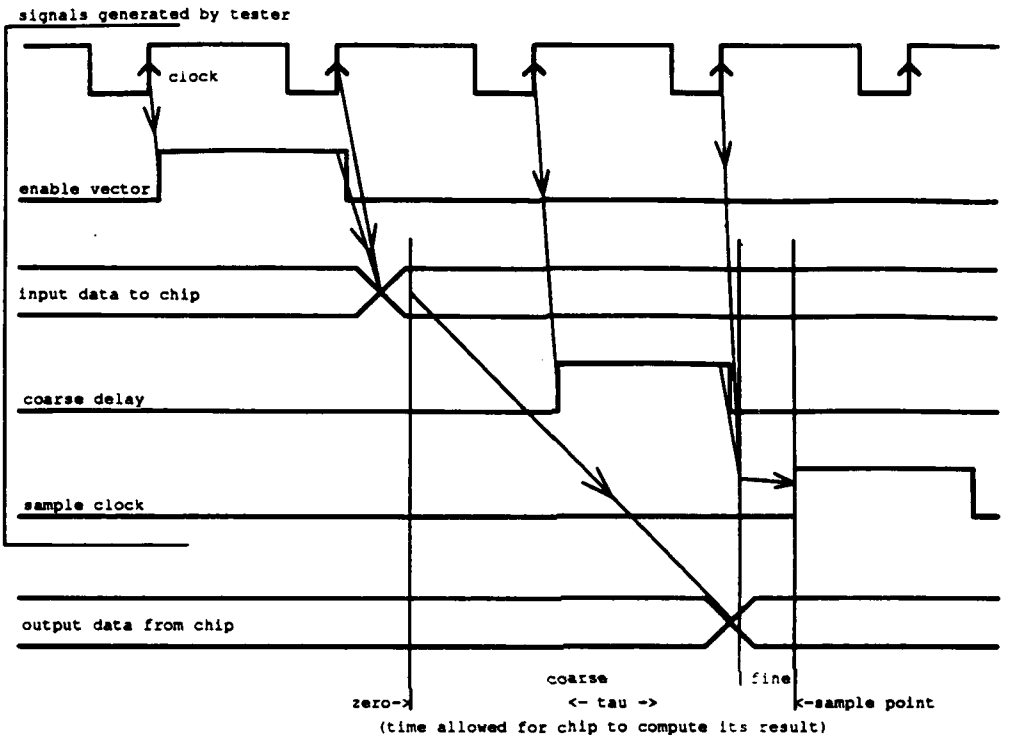


Figure 11: Tester timing

5.6 Pseudo speed-testing

Rudimentary speed testing can be performed by varying the timing of the sample clock. For example, the chip's inputs might be changed every microsecond, but the chip's outputs may be sampled tens of nanoseconds after the inputs arrive. This is not considered to be a true speed test, since internal state information may be changing more slowly than what can be sampled at the pins. See Figure 11.

For pseudo speed-testing, the sample clock's delay time τ must be adjustable. The sample clock signal is derived from the state machine clock when the enable vector signal (e.g. `Xselect<4>`, from Figure 10) is active. The variation is provided by two mechanisms: the coarse delay and the fine delay. When both delays are set to zero, the sample clock arrives simultaneously (within a few nanoseconds) with the current input vector. The coarse delay is a number of state-machine clock cycle times. The fine delay is composed as a series of gate propagation delays, which is less than a single clock cycle time.

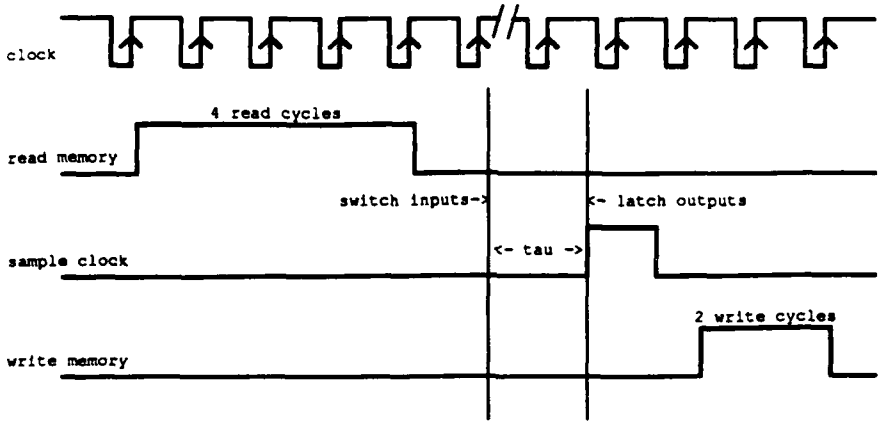


Figure 12: Tester timing for offline testing

5.7 The controller

Whether or not the tester is able to run tests offline, some sort of controller is needed. The controller provides handshake signals to the host bus and interfaces with the parallel port. It governs the data flow through to the test chip, and provides the timing for delivering the test vector and then latching the result. For chip testing offline from the CPU, the controller must do more. Data movement between the CPU and local memory may be performed when the tester is idle. Read and write cycles are performed during offline testing. Figure 12 describes the controller's timing for offline testing. Four consecutive memory locations are read. The input vector is delivered to the pins of the chip under test. The results are computed by the chip, and the controller moves them into the next two locations in memory.

The controller can execute a block of vectors stored in local memory, and then wait for the CPU to read back the results. It can execute this block once, or it can execute the block in an infinite loop. The latter feature is useful for displaying the test chip's signals on an oscilloscope.

In order to run tests offline from the CPU, the controller has an address counter, a loop counter and a delay counter. These counters are implemented using discrete TTL parts. In particular, 74LS693's are used which have buried registers (see [2]). These registers are used to store the counter's default value. This is useful for infinite loop mode, as the counters can be easily reloaded at the end of the loop in order to start over.

The address counter provides addresses to the RAMs in local memory. The loop counter keeps track of the size of the vector block, and it counts in step with the address counter.

When the count is zero, the block's end has been reached. The delay counter provides the coarse adjustment to the sample clock delay time. In the UW Chip Tester, the granularity of the coarse sample clock delay is 100 nanoseconds.

The controller can be implemented using a finite-state machine or a microprocessor to provide sequencing. FSMs are superior to MPUs in this case, where the control is simple and speed is critical. Typically, PALs are used as FSMs. However, the UW Chip Tester uses a Xilinx LCA as a FSM in its controller. There are several reasons why the LCA is an excellent choice. The number of control lines is large, and is beyond the range of most PALs. The reprogrammability of the LCA helps for implementing both the interactive and offline testing features of the tester. The fine (gate propagation) delay can be created using gates inside the LCA. This fine adjustment to the coarse delay time provides a wide range of possible delay times. Different versions of the controller are made available to the tester user, which have different sample clock delays and work with or without local memory.

5.8 Limitations

The UW Chip Tester is a low-cost, functional tester. Vectors are presented every few microseconds. Skew between test chip pins is on the order of nanoseconds. This is due to both the wire length and the characteristics of the Xilinx LCA, which has inherent skew between any pair of its output pins. Since the Xilinx LCA is CMOS, it has limited current sinking ability. The 3020 has only two power and two ground pins. If many of its outputs switch simultaneously, some signal degradation could occur. According to [1], a maximum of 32 outputs should switch simulateously. Since there are only 22 pin drivers per LCA, the problem is not severe.

6 Software test package

6.1 Low level interface package

At the lowest level, there is a library of functions and macros written in C for interfacing to the tester. The control register, on the interface card, is accessed to in order to initialize the LCAs and for running dynamic tests. The control register, the data path registers and the local memory (if installed) are simply memory-mapped into the Macintosh slot space of the interface card. The tester code is written to be slot-independent; at boot time, the Mac determines what cards are installed and what slots they occupy. The low level package queries the operating system to determine the slot number of the tester interface card. Refer to [3] for further discussion of NuBus slot addressing methods.

6.2 Test package

The test package contains a number of high-level constructs for creating test programs. There are two abstract data types that the test package uses: `PinType` and `SignalType`. `PinType` is a mapping between the physical chip pin and the socket. `SignalType` represents an array of Pins, which is convenient for describing busses. In this way, physical pins of the tester are mapped to integers. The maximum width of a Signal is 32 bits, which is the same size as a long integer on the Macintosh.

The physical test chip socket must have different logical pin numberings according to the footprint of the test chip. This mapping is provided by using different socket definition files.

There are several functions that the test package provides to interface with test programs. These functions are:

- `RunTest()`
- `NewSignal()`
- `Step()`
- `SetSignalData()`
- `SetSignalDirection()`
- `VerifySignalData()`

`RunTest()` is a function that initializes the tester and executes the test. The address of a user's function is passed as an argument; this function is called after hardware initialization. When the user's function returns, `RunTest()` performs cleanup, after which the program terminates.

`NewSignal()` creates and initializes a Signal. This function is called once per Signal at the beginning of the user's test code. The Pins controlled by the Signal are declared here, as well as the direction (input or output) of the Signal. `SetSignalData()` and `SetSignalDirection()` are used to change the data and direction of the Signal. The effects of these functions on a particular Signal persist until a subsequent call to either of these functions changes that Signal. Both these commands are buffered; no action takes place at the pins of the test chip until `Step()` is called.

`Step()` causes the tester to issue the current vector to the pins of the test chip, wait for the indicated delay time, and then latch the results. After `Step()` is called, the user can verify the outputs of the chip using the function `VerifySignalData()`.

For performing tests offline from the CPU, the following macros have been defined:

- `BEGIN_OFFLINE`
- `END_OFFLINE()`

These macros delimit an *OFFLINE block*. The commands in an offline block are executed as a batch. However, the style of programming closely resembles the interactive (CPU-driven) test. This allows the maximum portability of test programs between interactive and offline code. In reality, the OFFLINE block code is executed twice. On the first iteration, the local memory is set to the proper input vectors by executing `SetSignalData()` and `SetSignalDirection()` commands. Then the batch of vectors is executed offline. The test results are stored in local memory. Then the host code is executed in a second iteration, to perform all the `VerifySignalData()` commands. The user must ensure that the code block between the two macros is idempotent; all variables whose values change must be initialized inside the block before use. Functions with side effects (e.g. `scanf()`) must not execute during both passes of an OFFLINE block to ensure portability.

Further discussion of the test package is given in [4].

7 Future work

There is room for further experimentation with both the hardware and the software. One idea that we wish to implement is the four-vector queue for performing true speed testing. A preliminary study shows that this scheme would double the number of XC3020's required over the current scheme. The cost of the entire tester system would increase by about 80 per cent over the current tester system without memory.

On the software side, there is further system integration to be done. Soon, the test environment will include the use of Kyoto Common Lisp and the RNL simulation package. Since chip designers must use simulation to verify their designs prior to fabrication, it is advantageous for them to reuse the simulation code to test the chips. The overall software effort needed to perform chip testing is reduced.

8 Summary and conclusions

The UW Chip Tester achieves its design goals. Its design is a compromise between low cost and functionality. The use of Xilinx Logic Cell Arrays helped reduce costs, add flexibility, and reduce the physical dimensions of the tester. Its inexpensive, extensible design is useful for testing both static and dynamic chips. Its software library simplifies the design of test programs.

9 Acknowledgements

This project would not have come together without two key people: Carl Ebeling at UW and Ian Jones of Apple Computer. Carl provided the wisdom and experience from his involvement with the CMU Chip Tester, and proofread this paper. Ian shipped us the MacIIx development system from Apple, and provided software support. I thank Gaetano Borriello, Bill Barnard, and all the other members of the Laboratory for Integrated Systems at UW, for providing expert advice, tools, documentation, and participation in seminars and demonstrations of the tester. I also thank John Torode and his colleagues at IC Designs for providing chips to test.

References

[1] Alfke et al. *The Programmable Gate Array Data Book.* Copyright 1988 Xilinx, Inc.

[2] *The TTL Data Book, Volume 2.* Copyright 1985 Texas Instruments, Inc. ISBN 0-89512-096-8

[3] *Designing Cards and Drivers for Macintosh II and Macintosh SE.* Addison-Wesley Publishing Company, Inc. Copyright 1987 Apple Computer, Inc. ISBN 0-201-19256-8

[4] McKenzie, Neil. UW Chip Tester User Guide. University of Washington technical report [to be announced], December 1989.